МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное   
образовательное учреждение высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королёва»   
(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

ОТЧЁТ  
по циклу лабораторных работ   
по курсу «Системное программирование»

Выполнил:

Пороткин М.А.

гр.6303-090301D

Проверил:

к.т.н., доцент

Столбова А.А.

Самара 2021

Содержание

[Задание 3](#_Toc69996987)

[Приложение к заданию 4](#_Toc69996988)

[1 Лабораторная работа № 2 Проектирование системного программного обеспечения 6](#_Toc69996989)

[2 Лабораторная работа № 3 Создание сложной структуры данных 7](#_Toc69996990)

[3 Лабораторная работа № 4 Разработка системного программного обеспечения с использованием принципов SOLID и паттернов проектирования 8](#_Toc69996991)

[4 Лабораторная работа № 5 Вызов ассемблерных функций из языка высокого уровня 17](#_Toc69996992)

[5 Лабораторная работа № 6 Организация доступа к данным путем объектно-реляционного отображения 20](#_Toc69996993)

[6 Лабораторная работа № 7 Внедрение структурной обработки исключений 24](#_Toc69996994)

[7 Лабораторная работа № 8 Оценка эффективности функционирования системного программного обеспечения 26](#_Toc69996995)

[Приложение А Листинг класса AccessRecord 29](#_Toc69996996)

[Приложение Б Руководство пользователя 31](#_Toc69996997)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева» (Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

«УТВЕРЖДАЮ»

Руководитель лабораторных работ

Столбова А.А.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

ЗАДАНИЕ   
на цикл лабораторных работ по курсу «Системное программирование»  
студенту группы 6303-090301D

Пороткину Михаилу Александровичу

1. Исходные данные к лабораторной работе определяются согласно индивидуальному варианту: 9.
2. Перечень вопросов, подлежащих разработке в ходе выполнения лабораторных работ, приведен в Приложении к заданию.
3. Срок завершения цикла лабораторных работ: 31 мая 2021 г.
4. Дата выдачи задания: 18 февраля 2021 г.
5. График выполнения цикла лабораторных работ:

|  |  |  |  |
| --- | --- | --- | --- |
| Лабораторная  работа | Срок  выполнения | Итоги проверки | |
| Отметка  о выполнении | Подпись  преподавателя и дата |
| 1. Анализ требований к СПО. | 7.03.2021 |  |  |
| 2. Проектирование СПО. | 13.03.2021 |  |  |
| 3. Создание сложной структуры данных. | 21.03.2021 |  |  |
| 4. Разработка СПО с использованием принципов SOLID и паттернов проектирования. | 27.03.2021 |  |  |
| 5. Вызов ассемблерных функций из языка высокого уровня. | 4.04.2021 |  |  |
| 6. Организация доступа к данным путем объектно-реляционного отображения. | 10.04.2021 |  |  |
| 7. Внедрение структурной обработки исключений. | 18.04.2021 |  |  |
| 8. Оценка эффективности функционирования СПО. | 24.04.2021 |  |  |
| 9. Документирование СПО. | 30.04.2021 |  |  |

Задание принял к исполнению:

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / М.А. Пороткин

# ПРИЛОЖЕНИЕ к заданию

Цель цикла лабораторных работ – получение практических навыков проектирования, разработки, отладки, тестирования и документирования системного программного обеспечения (СПО).

Требования к информационному обеспечению:

1. СТО 02068410-004-2018 Общие требования к учебным текстовым документам [Текст]. – Самара: Самарский университет, 2018. – 36 с.
2. Г[ОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению](https://www.swrit.ru/doc/espd/19.505-79.pdf).

Функции, реализуемые разрабатываемым СПО:

1. СПО должно иметь возможность работы с файлом в формате Binary.
2. СПО должно иметь возможность сохранения файла и загрузки из файла введённых пользователем данных, а именно:

* Логин;
* Хэш-код пароля;
* Email.

1. СПО должно реализовывать разбор синтаксических конструкций, а именно:

* Оператор условного перехода if (<условие>) {<действие 1>} [else {<действие 2>}]
* Определить, какое из условий выполнится;

1. СПО должно реализовывать вызов низкоуровневых функций, а именно:

* Умножить два целочисленных значения с проверкой переполнения;

Требования к программному обеспечению:

Используемая версия операционной системы должна быть ОС Windows 7 и выше.

Требования к аппаратному обеспечению:

1. Процессор должен быть не ниже: Intel(R) Core(TM)
2. Установленная память (ОЗУ) должна быть не менее: 1 ГБ
3. Тип системы должен быть: 64-разрядная операционной системой.

Требования к языкам программирования и средам разработки СПО:

1. СПО должно быть разработано при использовании таких языков программирования, как С# и MSIL;
2. СПО должно быть разработано в среде разработки JetBrains Rider.

Требования к структуре СПО:

1. СПО должно включать исполняемый файл с расширением .exe;
2. СПО должно включать библиотеку, содержащую вызов низкоуровневых функций;
3. СПО должно включать библиотеку, содержащую бизнес-логику приложения.

Требования к способам организации диалога с пользователем:

1. СПО должно иметь графический пользовательский интерфейс, не нарушающий принятые подходы к проектированию интерфейсов;
2. Пользователь должен иметь возможность добавить, удалить, отредактировать запись данных.

Требования к качеству реализации функций:

1. СПО должно обеспечивать корректную работу с вводимыми данными, обеспечивать обработку исключительных ситуаций и формирование записи текущей работы, отображаемые на экране, в случае их возникновения;
2. Качество реализации функции должна обеспечивать безотказную работу программы.

Требования к быстродействию:

Не предъявляются.

Руководитель лабораторных работ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Столбова

Задание принял к исполнению: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / М.А. Пороткин

# Лабораторная работа № 2 Проектирование системного программного обеспечения

СПО включает в себя:

1. Porotkin\_SP\_9.exe – исполняемый файл, который содержит элементы графического пользовательского интерфейса, программный код для организации человеко-машинного взаимодействия и программный код, обращающийся к динамически подключаемым библиотекам.
2. LowLevelFunction.dll – библиотека, содержащая вызов низкоуровневых функций, соответствующих индивидуальному заданию. Данная библиотека включает:

* LowLevelMultiplication – функция, отвечающая за умножение двух целочисленных значений с проверкой переполнения.

1. Analyzer.dll – библиотека, содержащая бизнес-логику приложения. Данная библиотека включает:

* IfAnalyzer. Класс, который предназначен для выполнения анализа конструкции языка «if-else».

Схема СПО показана на рисунке 1.

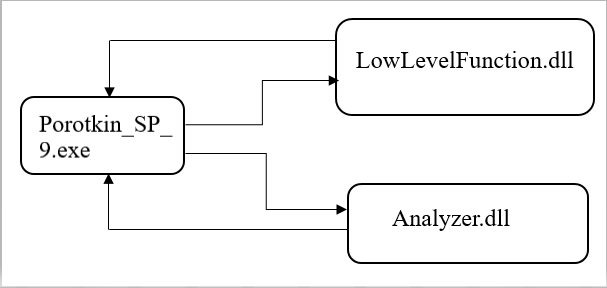


Рисунок 1 – Схема системного программного обеспечения

Требования к программному обеспечению:

Используемая версия операционной системы должна быть ОС Windows 7 и выше.

Требования к аппаратному обеспечению:

1. Процессор должен быть не ниже: Intel(R) Core(TM)
2. Установленная память (ОЗУ) должна быть не менее: 1 ГБ
3. Тип системы должен быть: 64-разрядная операционной системой.

# Лабораторная работа № 3 Создание сложной структуры данных

Для работы с файлами разработаем структуру данных, состоящую из класса «AccessRecord», код которого представлен в Приложении А.

Класс предназначен для хранения информации об одной записи о доступе и содержит такие поля, как:

* Id. Поле типа Guid, предназначенное для хранения уникального идентификатора записи о доступе;
* Login. Поле типа string, предназначенное для хранения логина;
* Passhash. Поле типа string предназначенное для хранения хэш-кода пароля.
* Email. Поле типа string предназначенное для хранения адреса электронной почты.

# Лабораторная работа № 4 Разработка системного программного обеспечения с использованием принципов SOLID и паттернов проектирования

При разработке СПО на языке высоко уровня будут применены современные подходы к разработке:

* Принцип проектирования SOLID
* Паттерн проектирования MVC

При разработке системного программного обеспечения нам необходимо определить поведение модели.

Определим основную логику в контроллерах, листинг которых приведен ниже.

public class AccessRecordController : Controller  
{  
 private readonly ApplicationDbContext \_db;  
 private readonly ILogger<AccessRecordController> \_logger;  
  
 public AccessRecordController(ApplicationDbContext db, ILogger<AccessRecordController> logger)  
 {  
 \_db = db;  
 \_logger = logger;  
 }  
  
  
 public IActionResult Index()  
 {  
 IEnumerable<AccessRecord> objList = \_db.AccessRecords;  
 return View(objList);  
 }  
  
 *// GET-Create* public IActionResult Create()  
 {  
 return View();  
 }  
  
 *// POST-Create* [HttpPost]  
 [ValidateAntiForgeryToken]  
 public IActionResult Create(AccessRecord obj)  
 {  
 \_db.AccessRecords.Add(obj);  
 \_db.SaveChanges();  
 \_logger.LogInformation($"New record with id = {obj.Id} has been created.");  
 return RedirectToAction("Index");  
 }  
  
 public IActionResult Update(Guid? id)  
 {  
 if (id == null)  
 {  
 return NotFound();  
 }  
  
 var result = \_db.AccessRecords.Find(id);  
 if (result == null)  
 {  
 return NotFound();  
 }  
  
 return View(result);  
 }  
  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public IActionResult Update(AccessRecord accessRecord)  
 {  
 \_db.AccessRecords.Update(accessRecord);  
 \_db.SaveChanges();  
 \_logger.LogInformation($"Record with id = {accessRecord.Id} has been updated.");  
 return RedirectToAction("Index");  
 }  
  
 public IActionResult Delete(Guid? id)  
 {  
 if (id == null)  
 {  
 return NotFound();  
 }  
  
 var result = \_db.AccessRecords.Find(id);  
 if (result == null)  
 {  
 return NotFound();  
 }  
  
 return View(result);  
 }  
  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public IActionResult DeletePost(Guid id)  
 {  
 var editRecord = \_db.AccessRecords.FirstOrDefault(element => element.Id == id);  
 if (editRecord != null)  
 {  
 \_db.AccessRecords.Remove(editRecord);  
 \_logger.LogInformation($"Record with id = {editRecord.Id} has been deleted.");  
 }  
  
 \_db.SaveChanges();  
 return RedirectToAction("Index");  
 }  
  
 public IActionResult GetFile()  
 {  
 FileStream file = System.IO.File.Create("test.bin");  
 BinaryWriter streamWriter = new BinaryWriter(file);  
 streamWriter.Write(\_db.AccessRecords.Count());  
 foreach (var accessRecord in \_db.AccessRecords)  
 {  
 streamWriter.Write(accessRecord.Id.ToString());  
 streamWriter.Write(accessRecord.Login);  
 streamWriter.Write(accessRecord.Passhash);  
 streamWriter.Write(accessRecord.Email);  
 }  
  
 streamWriter.Close();  
 \_logger.LogInformation($"Binary file has been exported.");  
 FileStream openedStream = System.IO.File.OpenRead("test.bin");  
 return File(openedStream, "application/binary", "test.bin");  
 }  
  
 public IActionResult Import()  
 {  
 return View();  
 }  
  
 [HttpPost]  
 public IActionResult ImportFromFile(IFormFile file)  
 {  
 BinaryReader binaryReader = new BinaryReader(file.OpenReadStream());  
 int count = binaryReader.ReadInt32();  
 for (int i = 0; i < count; i++)  
 {  
 string id = binaryReader.ReadString();  
 string login = binaryReader.ReadString();  
 string passhash = binaryReader.ReadString();  
 string email = binaryReader.ReadString();  
 AccessRecord record = new AccessRecord  
 {  
 Id = Guid.Parse(id), Login = login, Passhash = passhash, Email = email  
 };  
 \_logger.LogInformation($"File with name = {file.FileName} has been read.");  
 if (!\_db.AccessRecords.Contains(record))  
 {  
 \_db.Add(record);  
 }  
 }  
  
 binaryReader.Close();  
 \_db.SaveChanges();  
 return RedirectToAction("Index");  
 }

public class AnalyzerController : Controller  
{  
 private readonly ILogger<AnalyzerController> \_logger;  
  
 public AnalyzerController(ILogger<AnalyzerController> logger)  
 {  
 \_logger = logger;  
 }  
  
 public IActionResult Index()  
 {  
 return View();  
 }  
  
 public IActionResult Analyze(AnalyzerParams parameters)  
 {  
 var analyzer = new Analyzer.IfAnalyzer();  
 \_logger.LogDebug($"Inout if-statement: {parameters.IfBodyText}");  
 var result = analyzer.FindOutExecutedStatement(parameters.IfBodyText);  
 \_logger.LogDebug($"Analyzer output message: {result}");  
 return View(new AnalyzerResult(result));  
 }  
}

public class HomeController : Controller  
{  
 public HomeController(ILogger<HomeController> logger)  
 {  
 }  
  
 public IActionResult Index()  
 {  
 return View();  
 }  
  
 public IActionResult Privacy()  
 {  
 return View();  
 }  
  
 [ResponseCache(Duration = 0, Location = ResponseCacheLocation.**None**, NoStore = true)]  
 public IActionResult Error()  
 {  
 return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });  
 }  
}

public class LowLevelFunctionController : Controller  
{  
 private readonly ILogger<LowLevelFunctionController> \_logger;  
  
 public LowLevelFunctionController(ILogger<LowLevelFunctionController> logger)  
 {  
 \_logger = logger;  
 }  
  
 public IActionResult Index()  
 {  
 return View();  
 }  
  
 public IActionResult Calculate(LowLevelParams parameters)  
 {  
 var result = "";  
 try  
 {  
 result =  
 LowLevelFunction.  
 LowLevelFunction.  
 LowLevelMultiplication(parameters.First, parameters.Second).ToString();  
 }  
 catch (OverflowException e)  
 {  
 result = "The input values multiplication resulted in an overflow!";  
 }  
 \_logger.LogInformation(  
 $"The result of multiplication of {parameters.First} and {parameters.Second} is {result}");  
 return View(new LowLevelResult(result));  
 }  
}

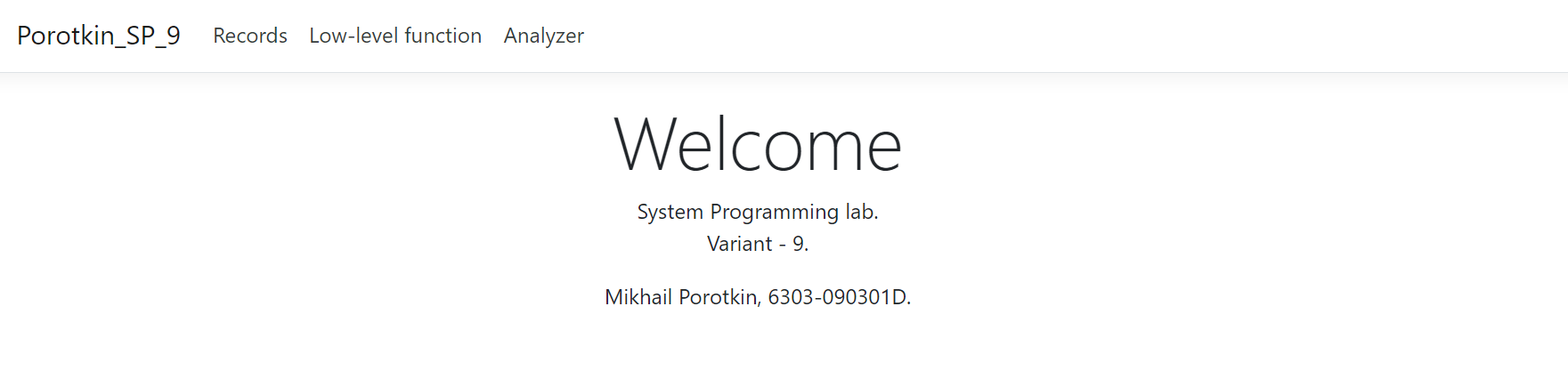


Рисунок 2 – Графическое представление программы

Для отображения результатов работы бизнес-логики описаны View, код которых приведен ниже.

@model Porotkin\_SP\_9.Models.AccessRecord  
  
<**form method**="post" **asp-action**="Create">  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Add Record</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <div class="col-4">  
 <**label asp-for**="Login"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Passhash"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Email"></**label**>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-4">  
 <**input asp-for**="Login" class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input asp-for**="Passhash" class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input** id="email" **asp-for**="Email" class="form-control"/>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-8 offset-2 row">  
 <div class="col">  
 <input type="submit" class="btn btn-info w-75" value="Create"/>  
 </div>  
 <div class="col">  
 <**a asp-action**="Index" class="btn btn-danger w-75">Back</**a**>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</**form**>  
<script>  
 $("#email").inputmask({alias: "email"});  
</script>

@model Porotkin\_SP\_9.Models.AccessRecord  
  
<**form method**="post" **asp-action**="DeletePost">  
 <**input asp-for**="Id" hidden/>  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Delete Record</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <div class="col-4">  
 <**label asp-for**="Login"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Passhash"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Email"></**label**>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-4">  
 <**input asp-for**="Login" disabled class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input asp-for**="Passhash" disabled class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input asp-for**="Email" disabled class="form-control"/>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-4"></div>  
 <div class="alert-danger col-4 mx-2">Do you really want to delete the record?</div>  
 <div class="col-4"></div>  
 </div>  
 <div class="form-group row">  
 <div class="col-8 offset-2 row">  
 <div class="col">  
 <input type="submit" class="btn btn-danger w-75" value="Yes"/>  
 </div>  
 <div class="col">  
 <**a asp-action**="Index" class="btn btn-success w-75">No</**a**>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</**form**>

@model Porotkin\_SP\_9.Models.AccessRecord  
  
<**form method**="post" enctype="multipart/form-data" **asp-action**="ImportFromFile">  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Import From File</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <input type="file" name="file" class="form-control">  
 </div>  
 <div class="form-group row">  
 <div class="col-8 offset-2 row">  
 <div class="col">  
 <input type="submit" class="btn btn-info w-75" value="Create"/>  
 </div>  
 <div class="col">  
 <**a asp-action**="Index" class="btn btn-danger w-75">Back</**a**>  
 </div>  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
</**form**>

@model IEnumerable<Porotkin\_SP\_9.Models.AccessRecord>  
  
<div class="container p-3">  
 <div class="row pt-4">  
 <div class="col-6">  
 <h2 class="text-primary">Access Records</h2>  
 </div>  
 <div class="col-2 text-right">  
 <**a asp-controller**="AccessRecord" **asp-action**="GetFile" class="btn btn-primary">Get File</**a**>  
 </div>  
 <div class="col-2 text-right">  
 <**a asp-controller**="AccessRecord" **asp-action**="Import" class="btn btn-primary">Import From File</**a**>  
 </div>  
 <div class="col-2 text-right">  
 <**a asp-controller**="AccessRecord" **asp-action**="Create" class="btn btn-primary">New Record</**a**>  
 </div>  
 </div>  
 <br/>  
  
 @if (Model.Count() > 0)  
 {  
 <table class="table table-bordered table-striped" style="width:100%">  
 <thead>  
 <tr>  
 <th>  
 Login  
 </th>  
 <th>  
 Password Hash  
 </th>  
 <th>  
 E-mail  
 </th>  
 <th>  
 Actions  
 </th>  
 </tr>  
 </thead>  
 <tbody>  
 @foreach (var item in Model)  
 {  
 <tr>  
 <td width="25%">@item.Login</td>  
 <td width="25%">@item.Passhash</td>  
 <td width="25%">@item.Email</td>  
 <td width="25%">  
 <**a asp-controller**="AccessRecord" **asp-action**="Update" **asp-route-id**="@item.Id" class="btn btn-primary mx-1">Update</**a**>  
 <**a asp-controller**="AccessRecord" **asp-action**="Delete" **asp-route-id**="@item.Id" class="btn btn-danger mx-1">Delete</**a**>  
 </td>  
 </tr>  
 }  
 </tbody>  
 </table>  
 }  
 else  
 {  
 <p>No items created yet</p>  
 }  
  
</div>

@model Porotkin\_SP\_9.Models.AccessRecord  
  
<**form method**="post" **asp-action**="Update">  
 <**input asp-for**="Id" hidden />  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Update Record</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <div class="col-4">  
 <**label asp-for**="Login"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Passhash"></**label**>  
 </div>  
 <div class="col-4">  
 <**label asp-for**="Email"></**label**>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-4">  
 <**input asp-for**="Login" class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input asp-for**="Passhash" class="form-control"/>  
 </div>  
 <div class="col-4">  
 <**input** id="email" **asp-for**="Email" class="form-control"/>  
 </div>  
 </div>  
 <div class="form-group row">  
 <div class="col-8 offset-2 row">  
 <div class="col">  
 <input type="submit" class="btn btn-info w-75" value="Update"/>  
 </div>  
 <div class="col">  
 <**a asp-action**="Index" class="btn btn-danger w-75">Back</**a**>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</**form**>  
<script>  
 $("#email").inputmask({alias: "email"});  
</script>

@model Porotkin\_SP\_9.Models.AnalyzerResult  
  
<div>  
 <div class="h2">  
 <h2>@Model.Result</h2>  
 </div>  
 <**a** class="btn btn-outline-info" **asp-action**="Index">Go Back</**a**>  
</div>

@model Porotkin\_SP\_9.Models.AnalyzerParams  
  
<**form method**="post" **asp-action**="Analyze">  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Check if-statement execution</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <div class="col-10">  
 <**input type**="text" **asp-for**="IfBodyText" class="form-control"/>  
 </div>  
 <div class="col-2">  
 <input type="submit" class="btn btn-success w-75" value="Analyze"/>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</**form**>

@{  
 ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
 <h1 class="display-4">Welcome</h1>  
 System Programming lab.  
 <p>Variant - 9.</p>  
 <p>Mikhail Porotkin, 6303-090301D.</p>  
</div>

@model Porotkin\_SP\_9.Models.LowLevelResult  
  
<div>  
 <h2>@Model.Result</h2>  
 <**a** class="btn btn-outline-info" **asp-action**="Index">Go Back</**a**>  
</div>

@model Porotkin\_SP\_9.Models.LowLevelParams  
  
<**form method**="post" **asp-action**="Calculate">  
 <div class="border p-3">  
 <div class="form-group row">  
 <h2 class="text-black-50 pl-3">Multiply two numbers</h2>  
 </div>  
 <div class="row">  
 <div class="col-12">  
 <div class="form-group row">  
 <div class="col-4">  
 <**input asp-for**="First" class="form-control"/>  
 </div>  
 <div>\*</div>  
 <div class="col-4">  
 <**input asp-for**="Second" class="form-control"/>  
 </div>  
 <div class="col">  
 <input type="submit" class="btn btn-success w-75" value="Calculate"/>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</**form**>

Пример работы программы представлен ниже на рисунках 3-4.

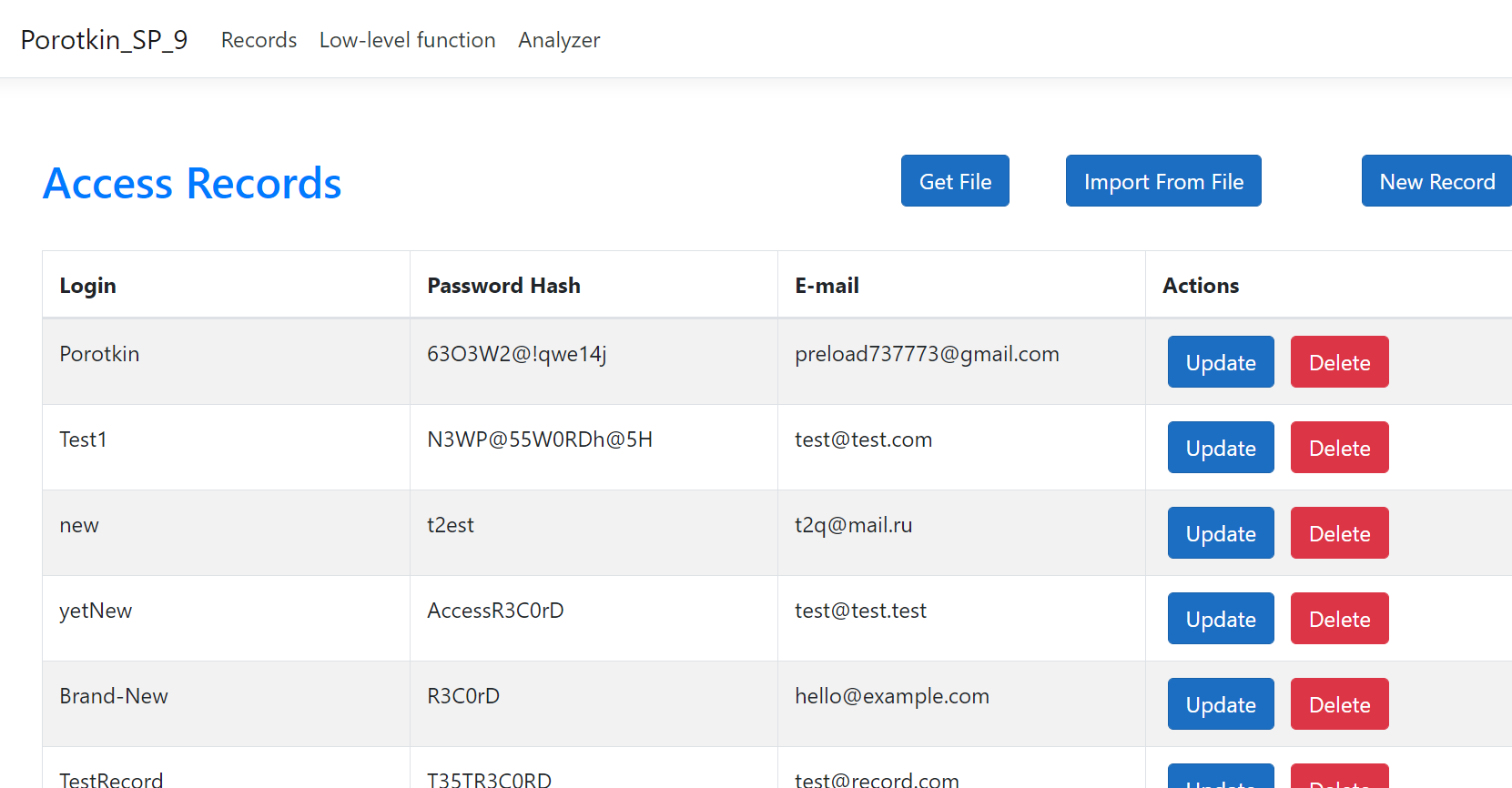


Рисунок 3 – Пример работы программы

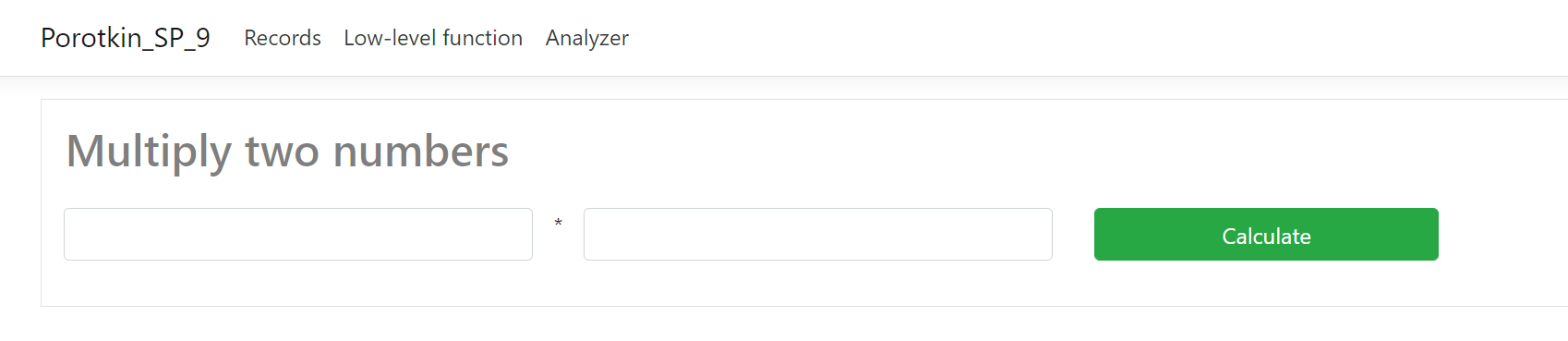


Рисунок 4 – Пример работы программы

# Лабораторная работа № 5 Вызов ассемблерных функций из языка высокого уровня

Разработаем организацию доступа к низкоуровневой функции с помощью низкоуровневых вставок на языке MSIL(CIL). Для создания динамически подключаемой библиотеки создадим дополнительное решение, которое формирует файл формата «.dll».

public class LowLevelFunction  
{  
 private delegate int Operation(int a, int b);  
   
 public static int LowLevelMultiplication(int a, int b)  
 {  
 Type[] OperationArgs = { typeof(int), typeof(int) };  
 DynamicMethod mul = new DynamicMethod("Mul", typeof(int), OperationArgs);  
 ILGenerator il = mul.GetILGenerator(256);  
 il.Emit(OpCodes.Ldarg\_0);  
 il.Emit(OpCodes.Ldarg\_1);  
 il.Emit(OpCodes.Mul\_Ovf);  
 il.Emit(OpCodes.Ret);  
 Operation result = (Operation)mul.CreateDelegate(typeof(Operation));  
 return result(a, b);  
 }  
}

var result = new LowLevelResult(  
 LowLevelFunction.  
 LowLevelFunction.  
 LowLevelMultiplication(parameters.First, parameters.Second));  
\_logger.LogInformation(  
 $"The result of multiplication of {parameters.First} and {parameters.Second} is {result}");  
return View(result);

Пример работы программы представлен на рисунках 5-7.

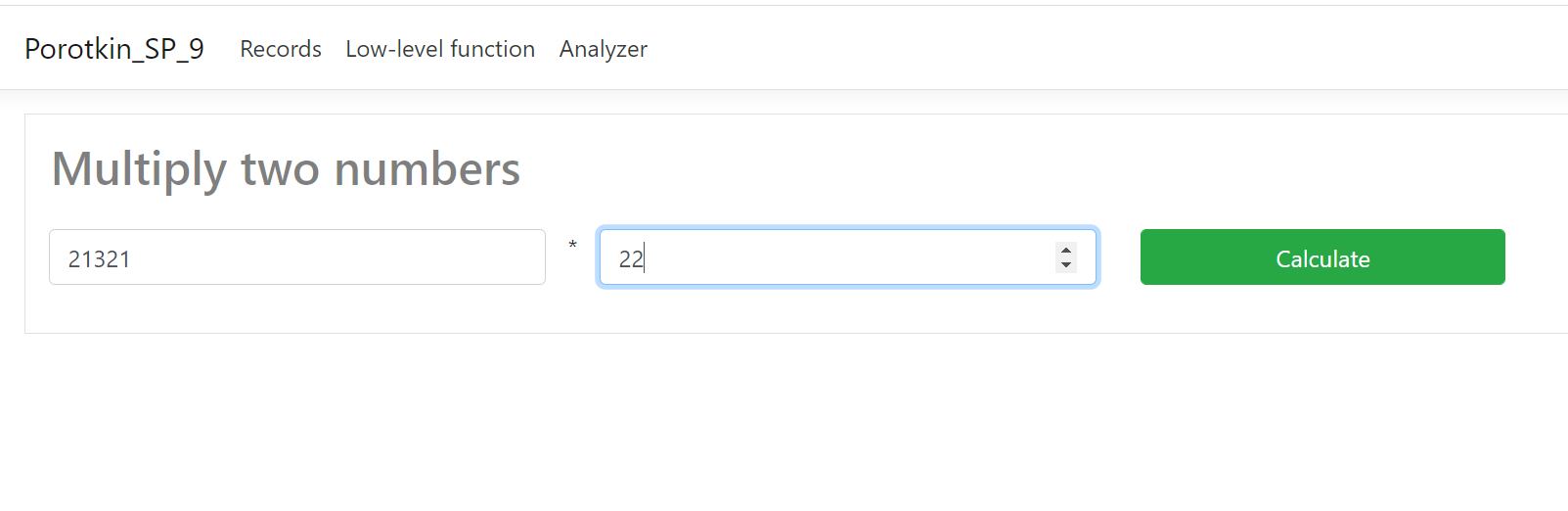


Рисунок 5 – Пример работы низкоуровневой функции

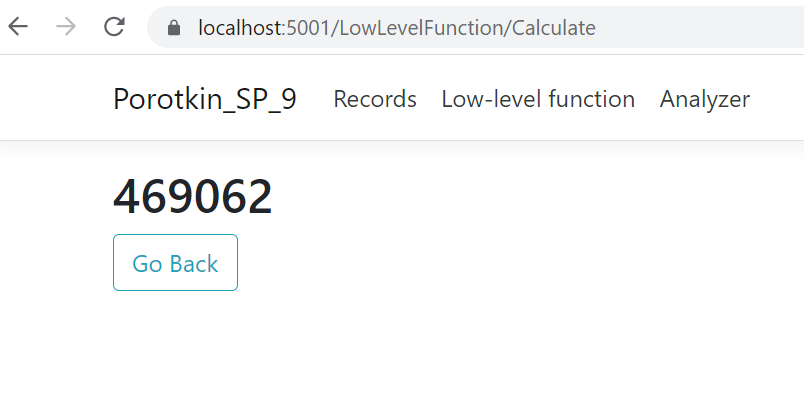


Рисунок 6 – Пример работы низкоуровневой функции

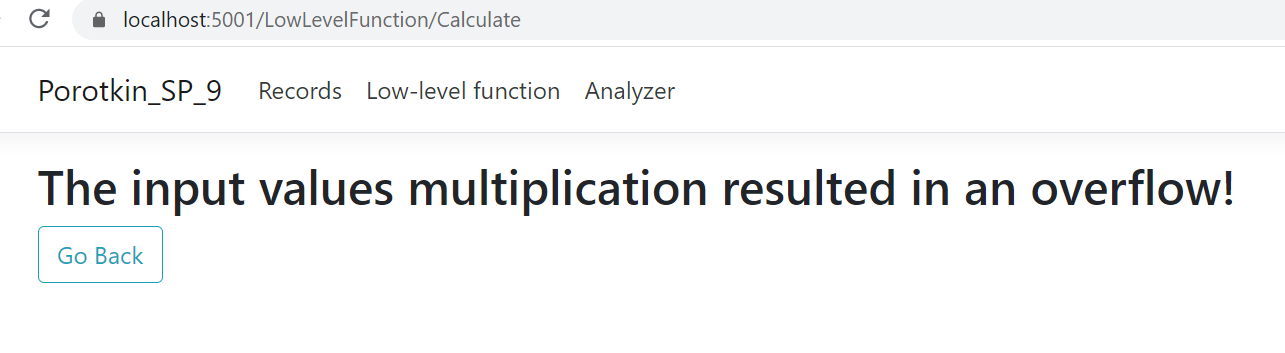


Рисунок 7 – Пример работы низкоуровневой функции

# Лабораторная работа № 6 Организация доступа к данным путем объектно-реляционного отображения

Для выполнения данной работы необходимо настроить непосредственно работу с базой данных. Для этого нужно подключиться к базе данных.

Графический интерфейс разрабатываемого СПО имеет вкладку для работы с записями базы данных.

На рисунке 9 приведен графический интерфейс приложения для работы с файлами из базы данных.

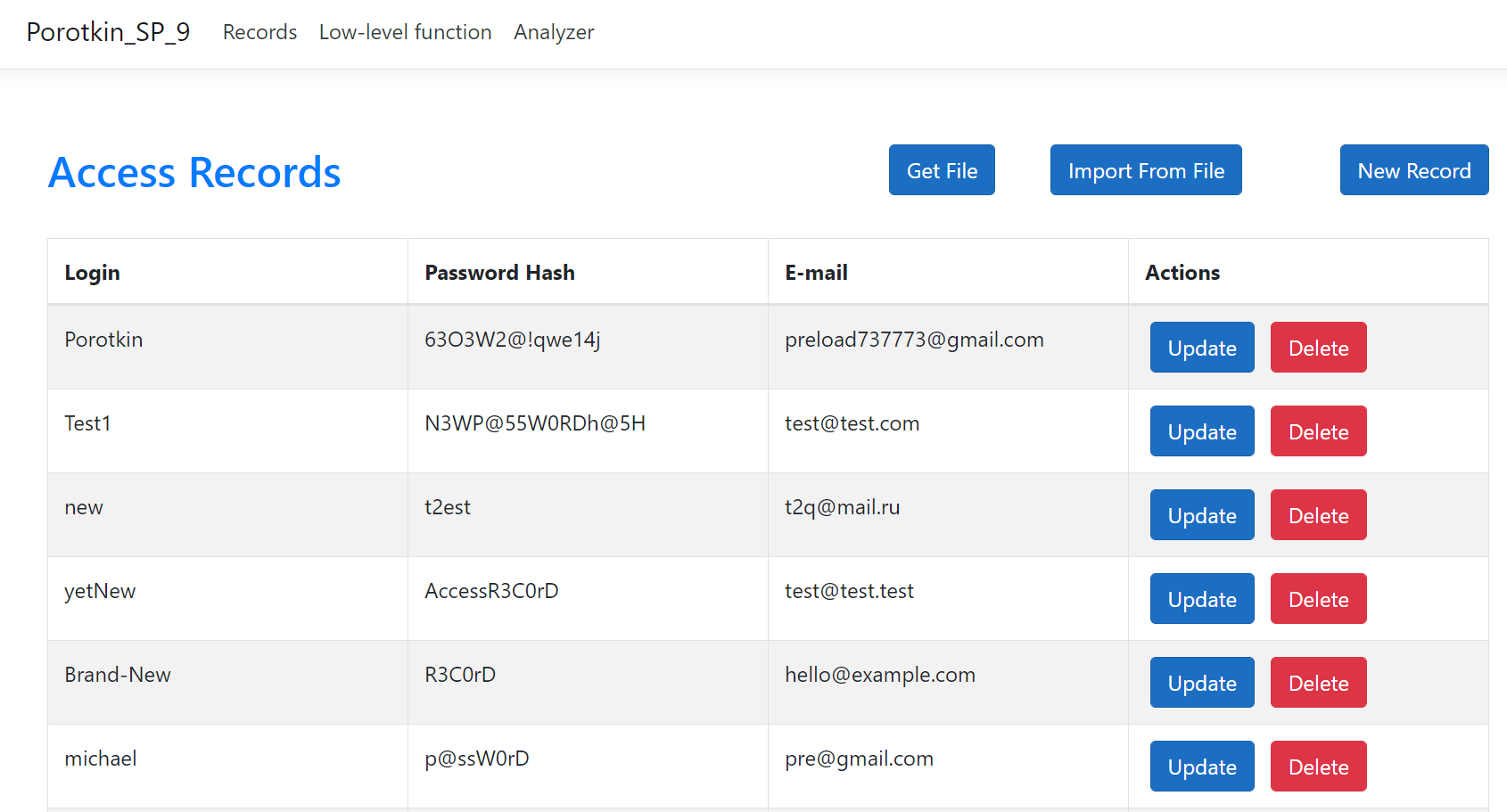


Рисунок 9 – Графический интерфейс приложения для работы с записями базы данных

Для реализации подключения к базе данных использован Entity Framework.

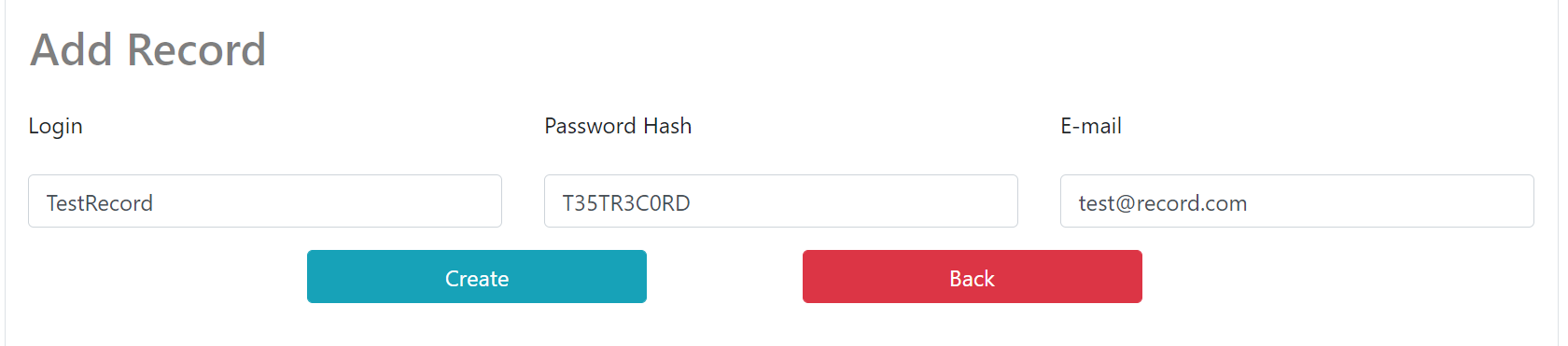


Рисунок 10 – Пример добавления элемента в базу данных

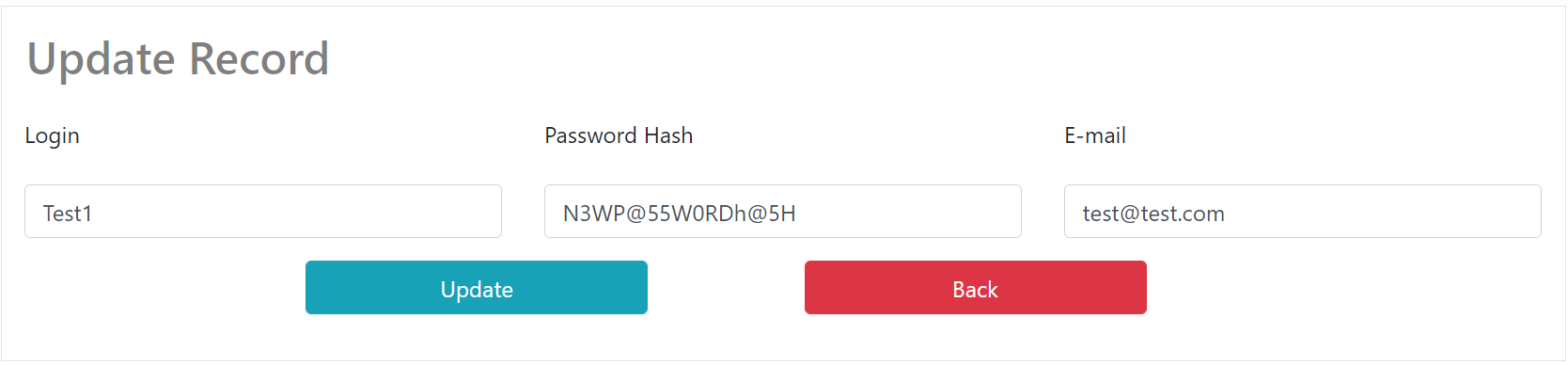


Рисунок 11 – Пример редактирования элемента базы данных

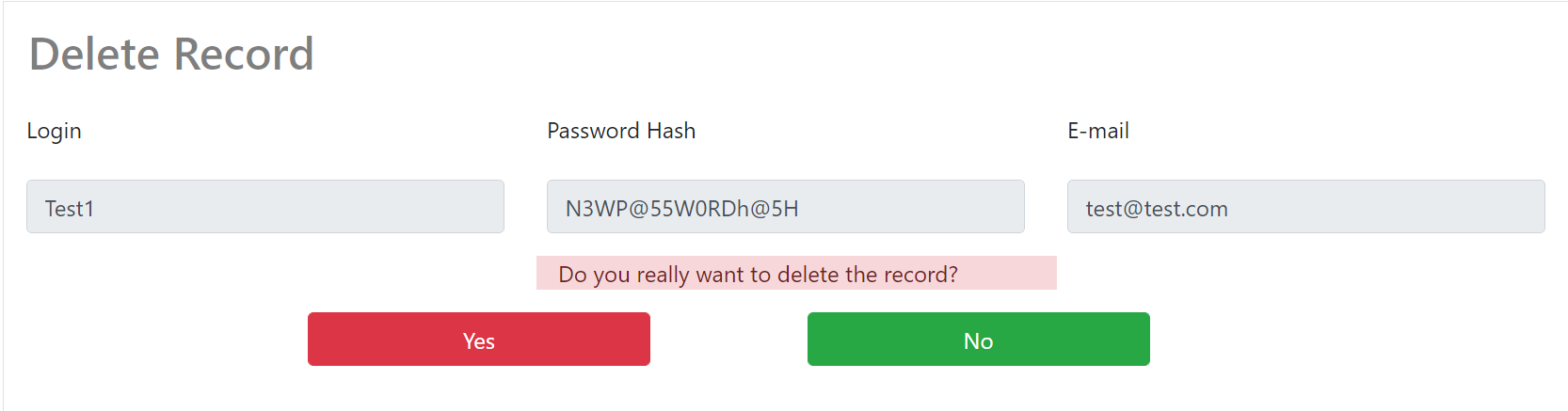


Рисунок 12 – Пример удаления элемента из базы данных

# Лабораторная работа № 7 Внедрение структурной обработки исключений

Разработаем для СПО структуру обработки исключительных ситуаций. Исключения будем обрабатывать с помощью встроенного оператора «try{} catch{}» C#. Код приведен ниже.

public IActionResult Calculate(LowLevelParams parameters)  
{  
 var result = "";  
 try  
 {  
 result =  
 LowLevelFunction.  
 LowLevelFunction.  
 LowLevelMultiplication(parameters.First, parameters.Second).ToString();  
 }  
 catch (OverflowException e)  
 {  
 result = "The input values multiplication resulted in an overflow!";  
 }  
 \_logger.LogInformation(  
 $"The result of multiplication of {parameters.First} and {parameters.Second} is {result}");  
 return View(new LowLevelResult(result));  
}

Пример обработки ошибки представлен ниже на рисунке 13.



Рисунок 13 – Пример обработки ошибки

# Лабораторная работа № 8 Оценка эффективности функционирования системного программного обеспечения

Оценим эффективность СПО и при необходимости проведём оптимизацию. С помощью dotTracer от JetBrains выявим использование памяти, оптимизации запросов и загруженность ЦП.

Запустим программу и проведём полный цикл работы программы. Рассмотрим дерево вызовов, представленное на рисунке 14. На нем мы видим время вызова, время и соотношения вызовов.

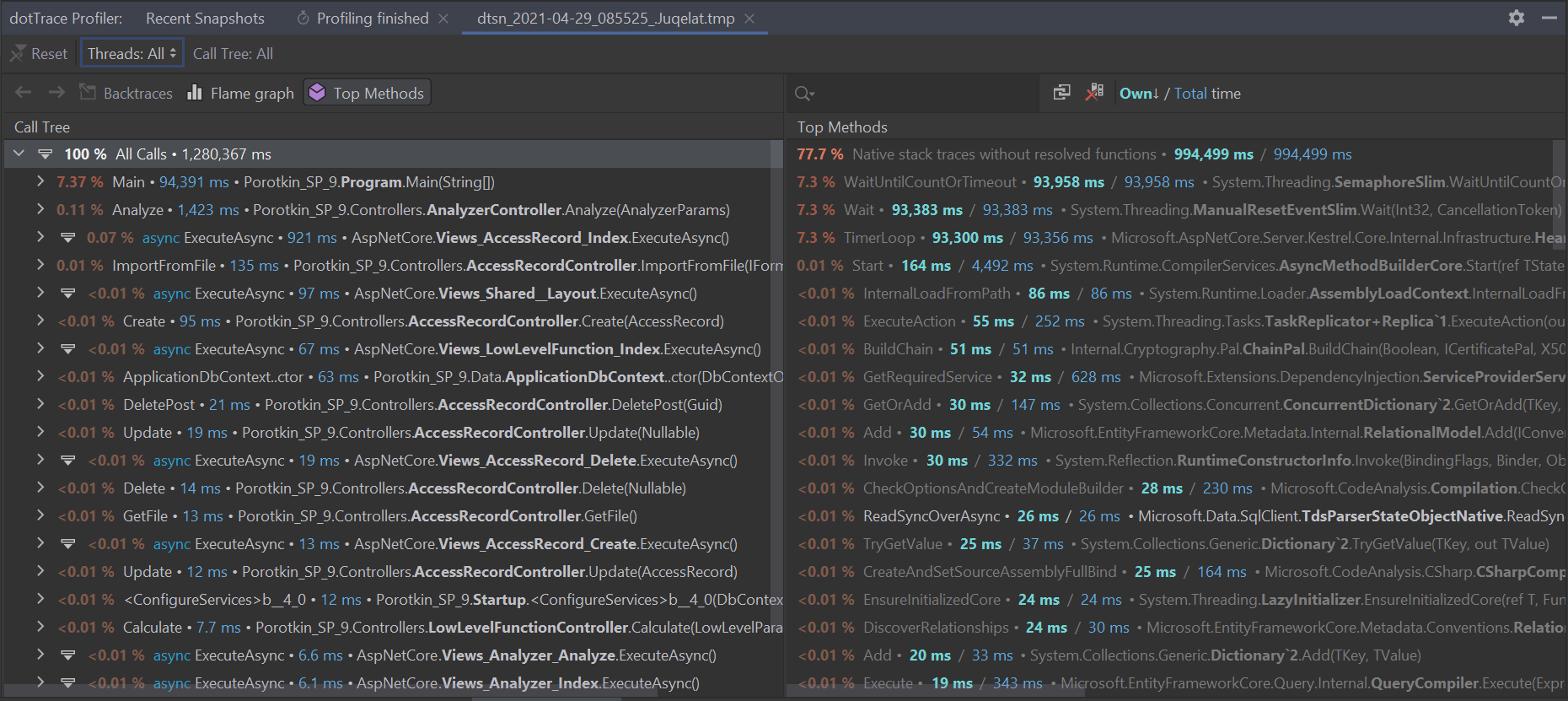


Рисунок 14 – Дерево вызовов

С помощью Dynamic Program Analysis проведем анализ на предмет присутствия утечек памяти в коде. На рисунке 15 показан результат работы утилиты.

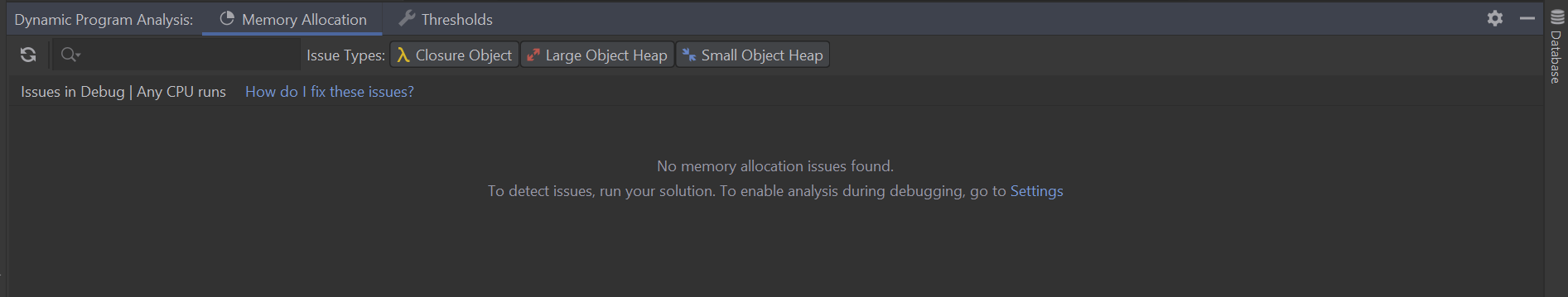


Рисунок 15 – Dynamic Program Analysis

С помощью встроенного в Visual Studio профилировщика производительности выясним недостатки в памяти программы. Получим результат, представленный на рисунке 16.

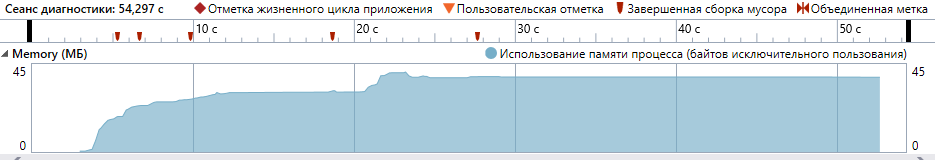


Рисунок 16 – Профилировщик производительности для памяти

При анализе памяти утечек памяти не обнаружено. Максимальный использованный объём памяти 45 МБ.

Запустим профилировщик производительности для ЦП. Данные приведены на рисунке 17.

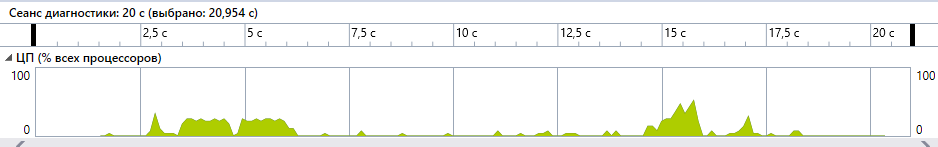


Рисунок 17 – Профилировщик производительности для ЦП

При анализе ЦП проблем не обнаружено. Максимальная нагрузка на ЦП составляет 50%. Осуществляется во время работы с файлами: открытие и сохранение.

Вывод: на основание полученных результатов исследований программа является эффективной. СПО в оптимизации не нуждается.

# ПРИЛОЖЕНИЕ А Листинг класса AccessRecord

using System;  
using System.ComponentModel;  
using System.ComponentModel.DataAnnotations;  
  
namespace Porotkin\_SP\_9.Models  
{  
 public class AccessRecord  
 {  
 [Key] public Guid Id { get; set; }  
 [DisplayName("Login")] public string Login { get; set; }  
 [DisplayName("Password Hash")] public string Passhash { get; set; }  
 [DisplayName("E-mail")] public string Email { get; set; }  
 }  
}

# ПРИЛОЖЕНИЕ Б Руководство пользователя

Работа с записями базы данных и с бинарными файлами:

1. Бинарный файл можно загрузить на сайт по кнопке “Import from file” вкладки “Records”. Все записи, содержащиеся в папке, но не существующие в базе данных, будут в нее добавлены.
2. Файл с текущими записями можно получить, нажав на кнопку “Get file”.
3. Для добавления записи в базу данных необходимо нажать кнопку «New Record» и заполнить данные о записи в соответствующей форме.
4. Для редактирования необходимо нажать кнопку «Update» на нужной записи и заполнить данные о записи в соответствующей форме.
5. Для удаления записи из таблицы необходимо нажать “Delete” на нужной записи и подтвердить удаление в соответствующей форме.

Ниже приведен работы с записями базы данных.

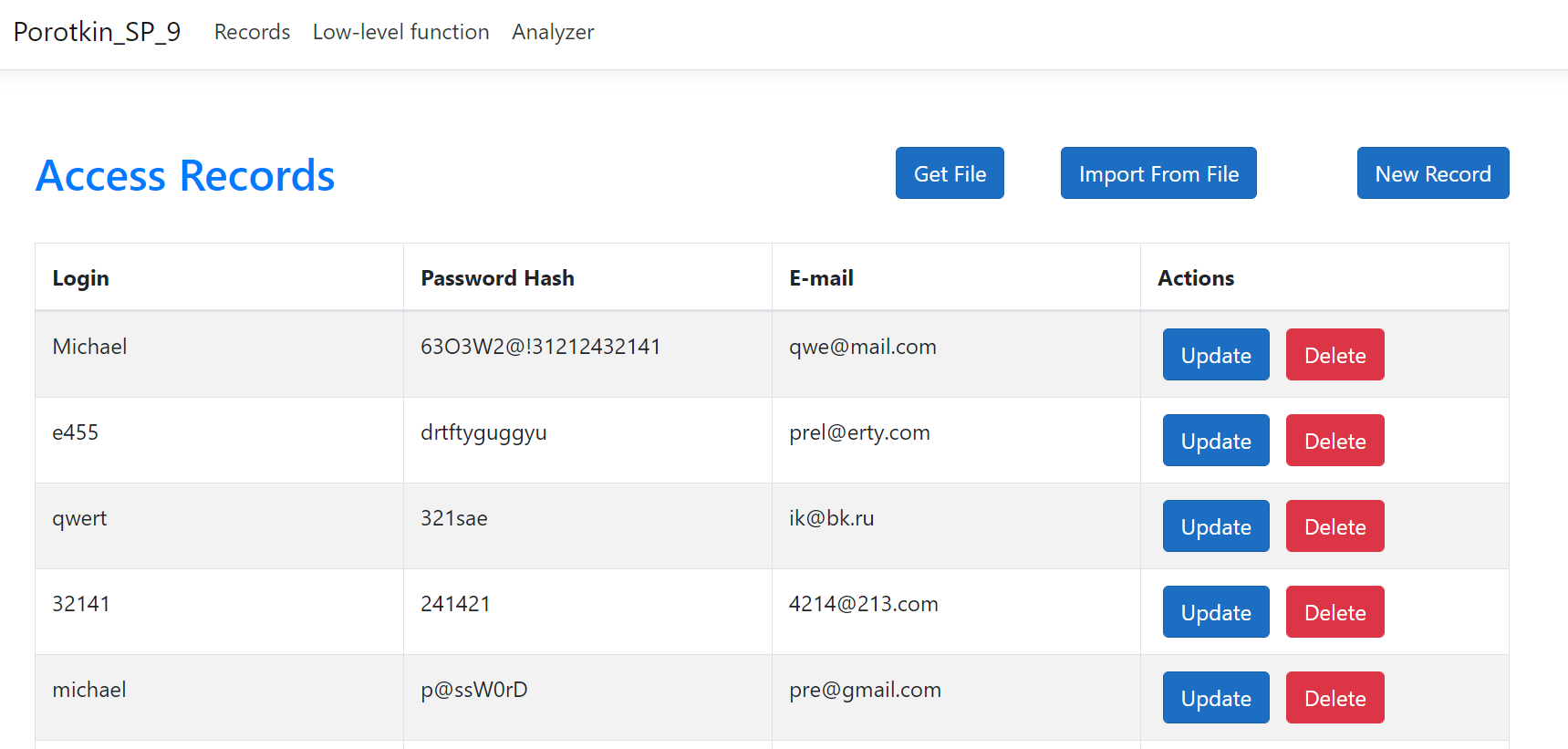


Рисунок Б.1 – Форма вывода всех записей на экран

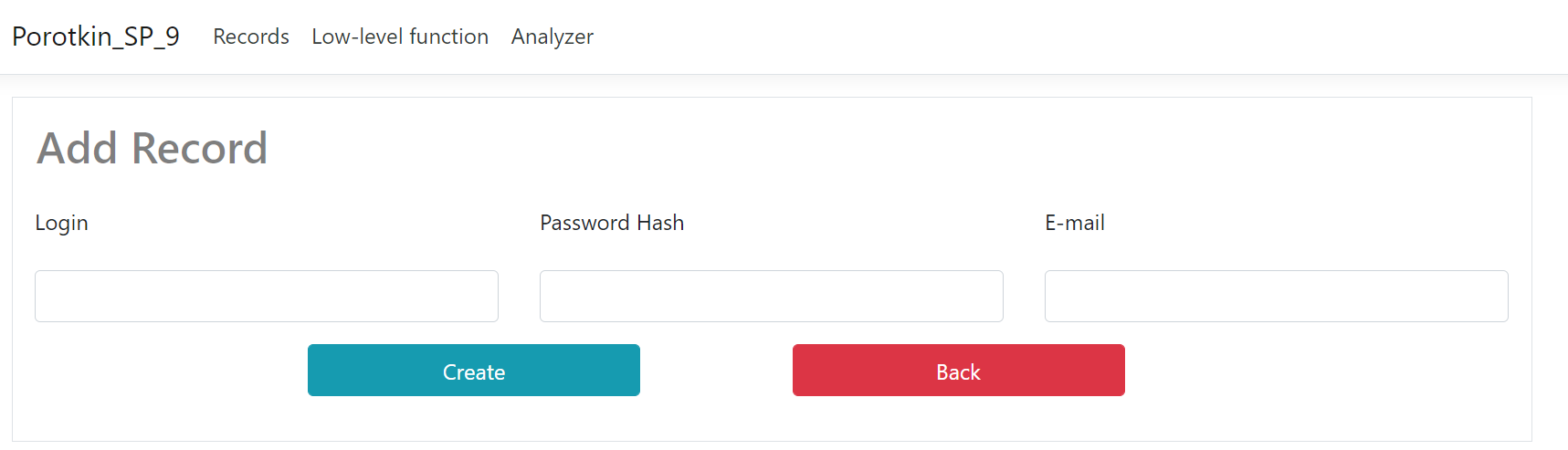


Рисунок Б.2 – Форма добавления новой записи

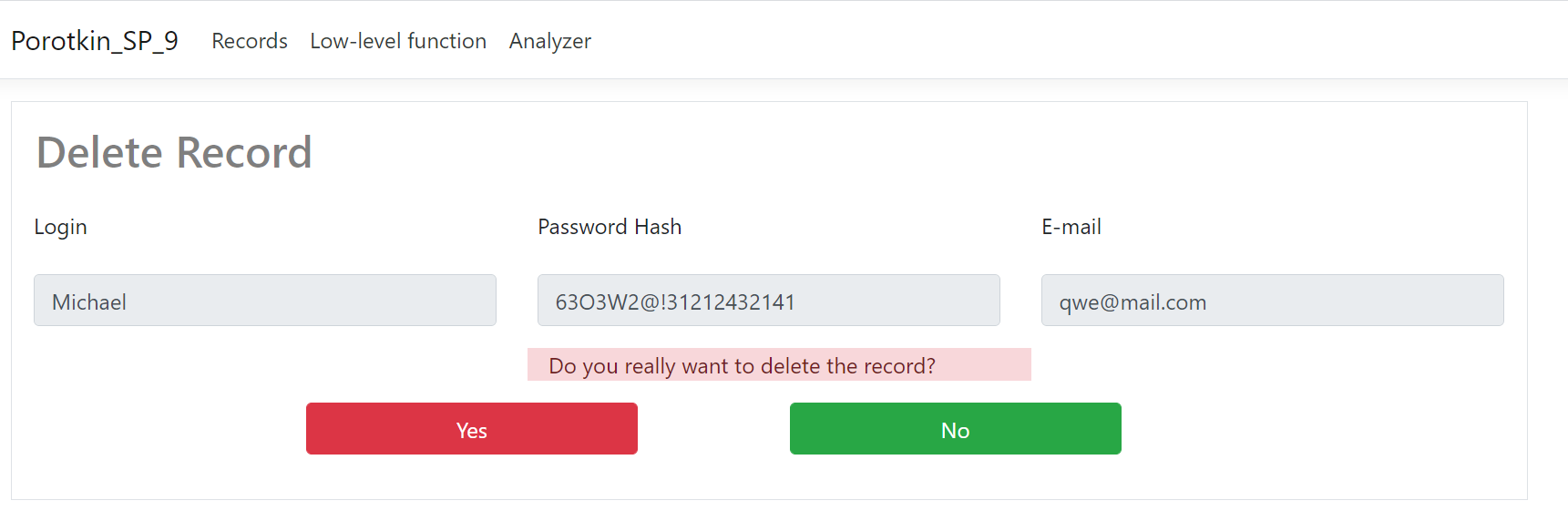


Рисунок Б.3 – Форма удаления записи

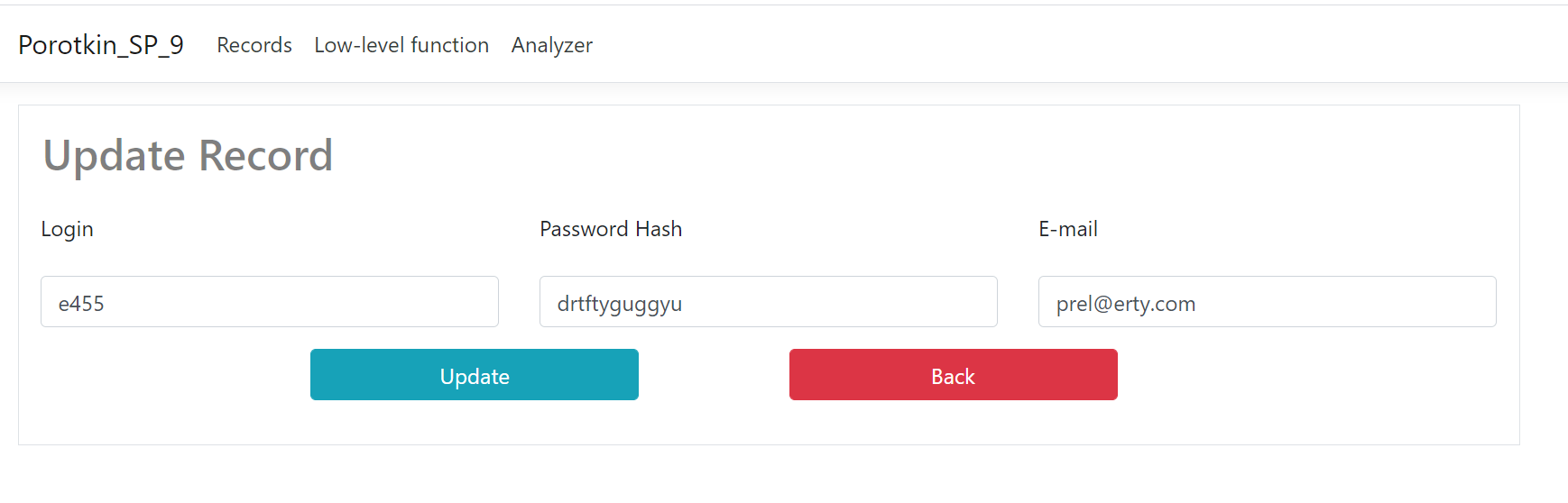


Рисунок Б.4 – Форма редактирования записи

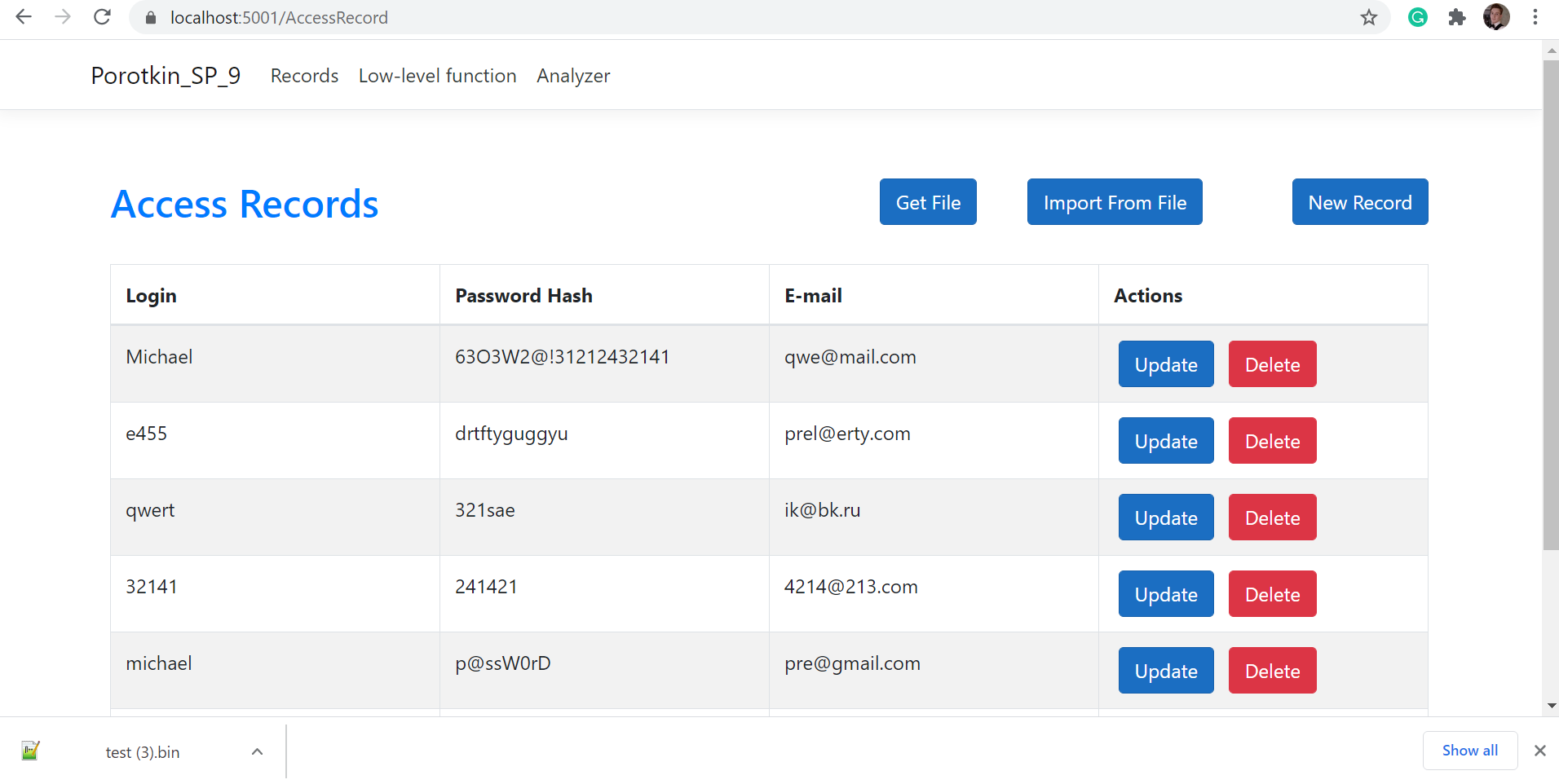


Рисунок Б.5 – Сохранение бинарного файла через кнопку “Get File”

Работа с низкоуровневой функцией:

1. Перейдите на вкладку «Low-level Function».
2. Введите два числа, с которыми будет проводиться операция умножения с проверкой переполнения.
3. Нажмите кнопку «Calculate».
4. В качестве положительного результата сравнения будет выведено произведение введенных чисел, в качестве неудачи «The input values multiplication resulted in an overflow».

Пример работы низкоуровневой функции сравнения чисел приведен ниже.

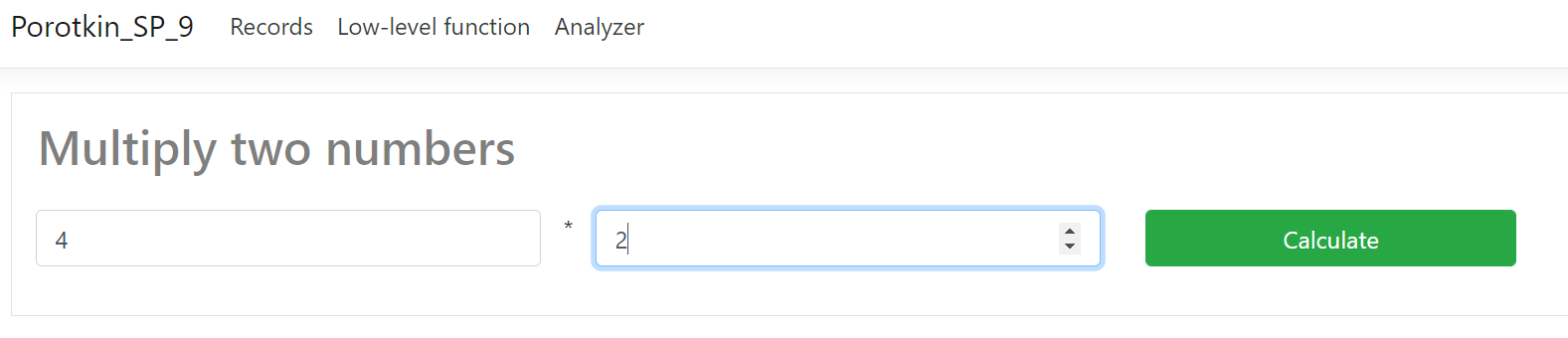


Рисунок Б.6 – Пример умножения двух чисел

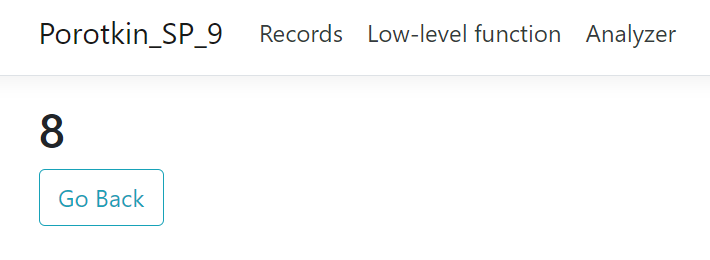


Рисунок Б.7 – Пример умножения двух чисел

Работа с конструкцией языка С#:

1. Выберите вкладку «Analyzer».
2. Введите конструкцию в поле ввода.
3. Нажмите кнопку «Analyze».
4. Если присутствуют ошибки и анализ не прошел успешно, будет выведено сообщение о том, где произошла ошибка.

Пример анализа конструкции языка C# приведен ниже:

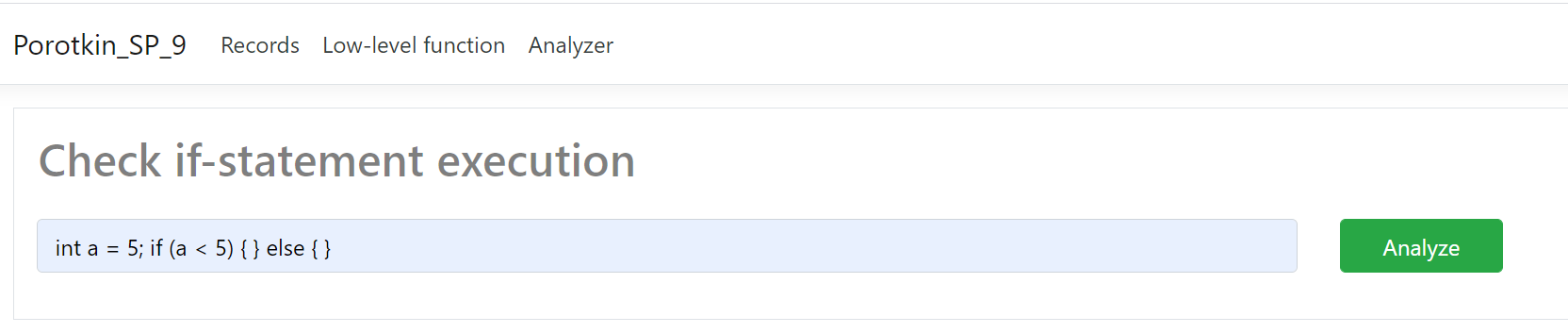


Рисунок Б.8 – Представление работы с конструкцией языка C#



Рисунок Б.9 – Представление работы с конструкцией языка C#

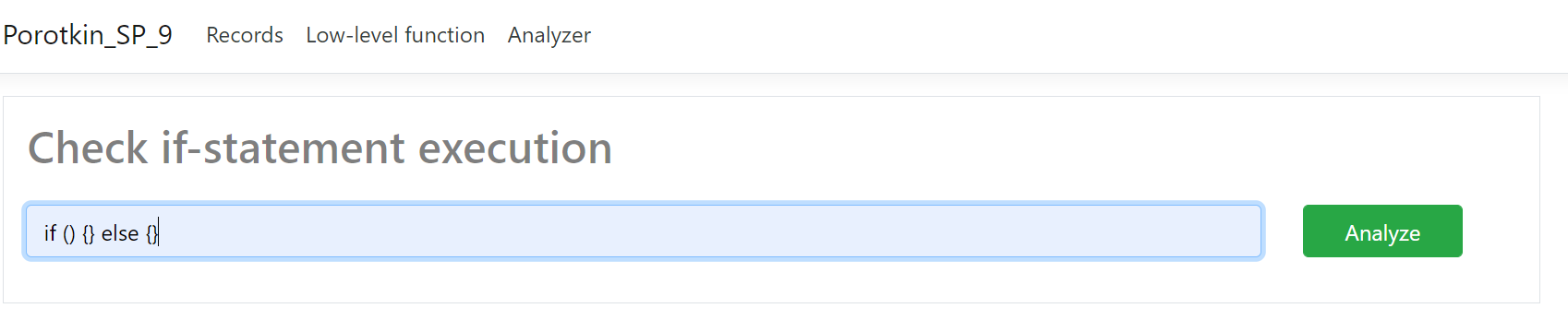


Рисунок Б.10 – Представление работы с конструкцией языка C#

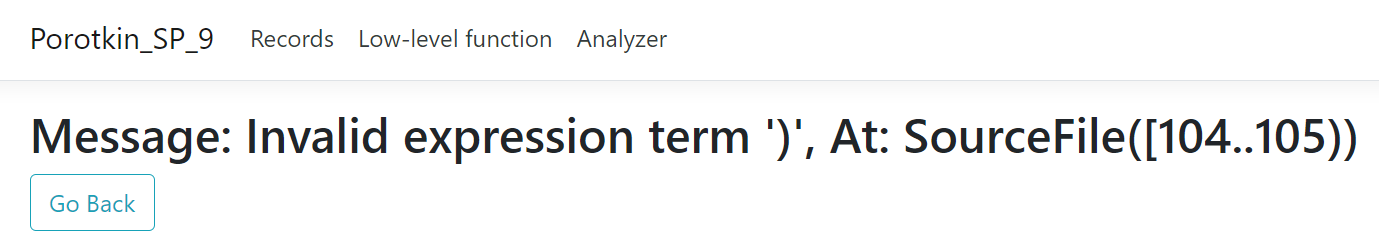


Рисунок Б.11 – Представление работы с конструкцией языка C#